Hacettepe University Department of Computer Engineering

Name & Surname	: Baturalp Furkan KARAMUS
Identity Number	: 21527137
Course	: BBM418 - Computer Vision Laboratory
Assignment	: Assignment2
Advisors	: Assoc. Prof. Dr. Nazlı İKİZLER CİNBİŞ, TA. Özge YALÇINKAYA
Due	: 06.05.19
E-mail	: b21527137@cs.hacettepe.edu.tr

1 Introduction

1.1 Problem

In this assignment, we will analyze 3 different method for image detection based algorithm depends their accuracy on detect images.These methods use an algorithm which is Convolutional Neural Networks. While using this algorithm we will use pre trained model which is VGG16 model for train images. The problem is analysing this algorithm and compare which one is the best solution for us.

This report contains all the steps involved in the process, as well as how the problems encountered in these stages are optimized and relevant.

2 Part 1

In this section i use Convolutional Neural Network algorithm for extract features from image. While doing this vgg16 pre trained model used. This is trained by someone else(ImageNet) previously by other projects. We use this model because this model prevent us build our CNN again.

Purpose is apply this model all images in test and train set. Then extract feature vector which fc7 after ReLu. These are 4096 dimensional vectors and include informations about image specifications.

After that i use Support Vector Machine for classify these features and calculate accuracy of predictions.While doing this i use linear kernel and one-to-multi approach.

2.1 Pre Trained VGG-16 Model

Simply put, a pre-trained model is a model created by some one else to solve a similar problem. Instead of building a model from scratch to solve a similar problem, you use the model trained on other problem as a starting point.

For example, if you want to build a self learning car. You can spend years to build a decent image recognition algorithm from scratch or you can take inception model (a pre-trained model) from Google which was built on ImageNet data to identify images in those pictures.

A pre-trained model may not be 100% accurate in your application, but it saves huge efforts required to re-invent the wheel

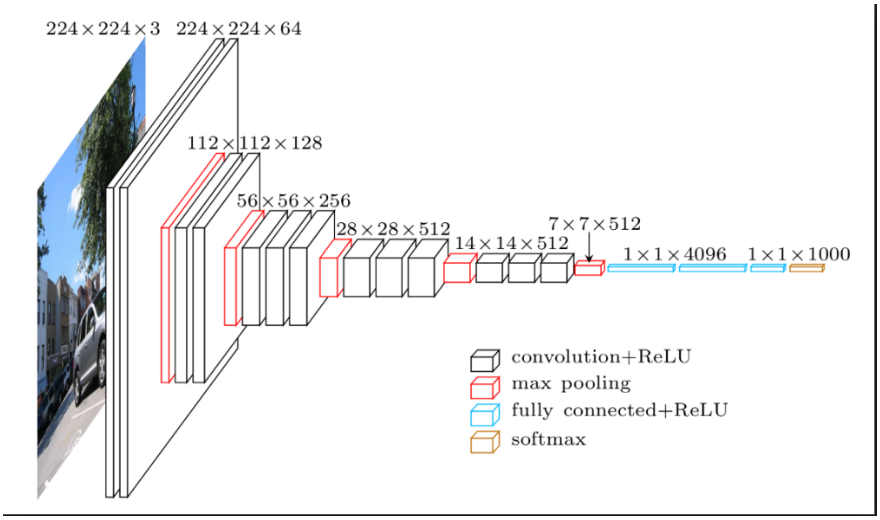

Figure 1: Basically vgg16 pre trained model.

2.2 Support Vector Machines

A hyper plane in an n-D feature space can be represented by the following equation:

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b = \sum_{i=1}^{n} x_i w_i + b = 0$$

Dividing by $||\mathbf{w}||$, we get

$$\frac{\mathbf{x}^T\mathbf{w}}{||\mathbf{w}||} = P_{\mathbf{w}}(\mathbf{x}) = -\frac{b}{||\mathbf{w}||}$$

indicating that the projection of any point $\mathbf{x}$ on the plane onto the vector $\mathbf{w}$ is always $-b/||\mathbf{w}||$, i.e., $\mathbf{w}$ is the normal direction of the plane, and $|b|/||\mathbf{w}||$ is the distance from the origin to the plane. Note that the equation of the hyper plane is not unique. $c\,f(\mathbf{x}) = 0$ represents the same plane for any c.

The n-D space is partitioned into two regions by the plane. Specifically, we define a mapping function $y = sign(f(\mathbf{x})) \in \{1, -1\}$,

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b = \begin{cases} > 0, & y = sign(f(\mathbf{x})) = 1,\ \mathbf{x} \in P \\ < 0, & y = sign(f(\mathbf{x})) = -1,\ \mathbf{x} \in N \end{cases}$$

Any point $\mathbf{x} \in P$ on the positive side of the plane is mapped to 1, while any point $\mathbf{x} \in N$ on the negative side is mapped to -1. A point $\mathbf{x}$ of unknown class will be classified to P if $f(\mathbf{x}) > 0$, or N if $f(\mathbf{x}) < 0$.

Example: A straight line in 2D space $\mathbf{x} = [x_1, x_2]^T$ described by the following equation:

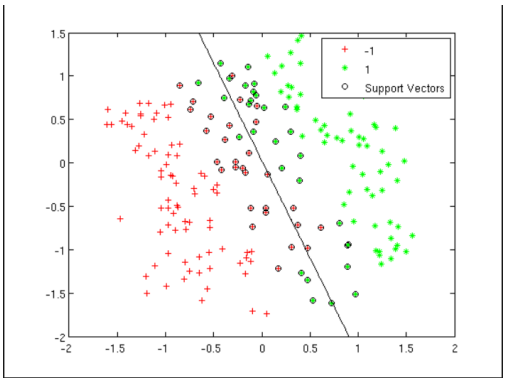

Figure 2: Support Vector Machine.

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{w} + b = [x_1, x_2]\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} + b = [x_1, x_2]\begin{bmatrix} 1 \\ 2 \end{bmatrix} - 1 = x_1 + 2x_2 - 1 = 0$$

devides the 2D plane into two halves. The distance between the origin and the line is

$$\frac{|b|}{||\mathbf{w}||} = \frac{1}{\sqrt{w_1^2 + w_2^2}} = \frac{1}{\sqrt{5}} = 0.447$$

Consider three points:

- $\mathbf{x}_0 = [0.5,\ 0.25]^T$, $f(\mathbf{x}_0) = 0.5 + 2 \times 0.25 - 1 = 0$, i.e., $\mathbf{x}_0$ is on the plane;
- $\mathbf{x}_1 = [1,\ 0.25]^T$, $f(\mathbf{x}_1) = 1 + 2 \times 0.25 - 1 = 0.5 > 0$, i.e., $\mathbf{x}_1$ is above the straight line;
- $\mathbf{x}_2 = [0.5,\ 0]^T$, $f(\mathbf{x}_2) = 0.5 + 2 \times 0 - 1 = -0.5 < 0$, i.e., $\mathbf{x}_2$ is below the straight line.

Software Usage

I use basic vgg16 model for extract features. While doing this torch library was used. After that classify images with SVM classifier which implemented with scikit learn library.

- Read dataset and initiliaze transform parameters.
- Create dataloader objects and decide batch size.
- Create vgg16 model which pre-tranined true.
- Pop the fc8 item on classifier.
- Put all images in vgg16 model in evaluation mode then extract features.
- Store all features in array
- Create support vector machine object.
- Fit classifier object with train labels and train features.
- Then predict with test features.
- Finally calculate accuracy with accuracy_score function with metrics library.
- Draw confusion matrix.

Experiment Results
In this part i will briefly explain my test results with my own comments. This table shows up two different

Table 1: Part 1 results

Part		
Batch Size	Correct	Accuracy
16	214	%85.6
32	208	%83.2

tests.One of them is tested with batch size 16 other is 32.This model have high weight size so it nearly run slow.

When i increase batch size program run faster but accuracy decrease.These are confusion matrices for this two test.

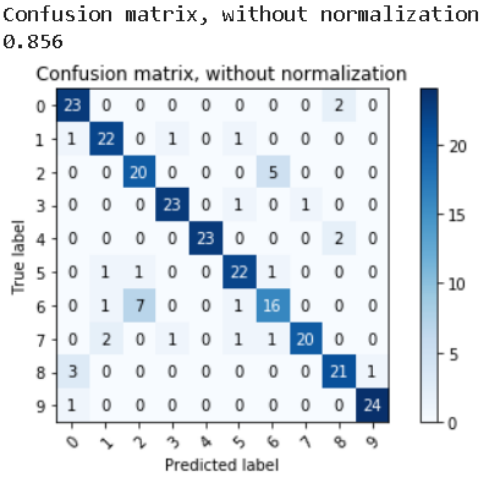

Figure 3: Confusion Matrix For Batch Size 16

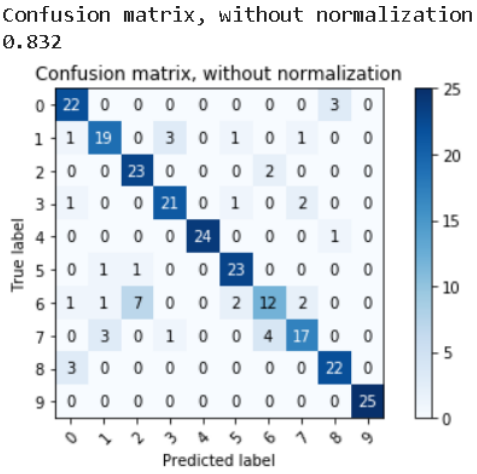

Figure 4: Confusion Matrix For Batch Size 32.

These matrices shows us which class have how many correct answer.According to these we can say that mostly classes have good accuracy but in living room and restaurant images have lowest accuracy than other classes. I think this is related to the similarity of two class feature vectors.

Implementation Details

In this part there is some explanation about test steps:

1-Create Model:
In this step i create vgg16 pretrained model and pop the last layer for extract fc7 layer.
2-Get SVM object:
In this step i create svm object with scikitlearn svm library. Then initiliaze parameters. I use linear svm.
3-Extract Features:
In this step iterate over on dataloaders and add every images in model as input. Take features and append them all to the list and append their correspond labels to other list. Before appending i convert them to the numpy array. After them prepare this list for using svm classification.
4-Classify:
Final step we compare test image and predictions. Create prediction array with scikit learn predict function. After that calculate accuracy score with correct labels.

3 PART 2

In this part i will fine-tune the given pre-trained VGG-16 network which loaded all the weights from the torch library. This network is trained on ImageNet.After that freezed all the layers before training the network, except the FC layers. Consequently, the gradients will not be calculated for the layers except the ones that we are going to update (FC layers) over the pre-trained weights. However, since the number of classes is different for our dataset, we should modify the last layer of the network, FC-8, which the probabilities will be calculated on.

3.1 Convolutional Neural Networks

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery.

CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually refer to fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks make them prone to overfitting data. Typical ways of regularization includes adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

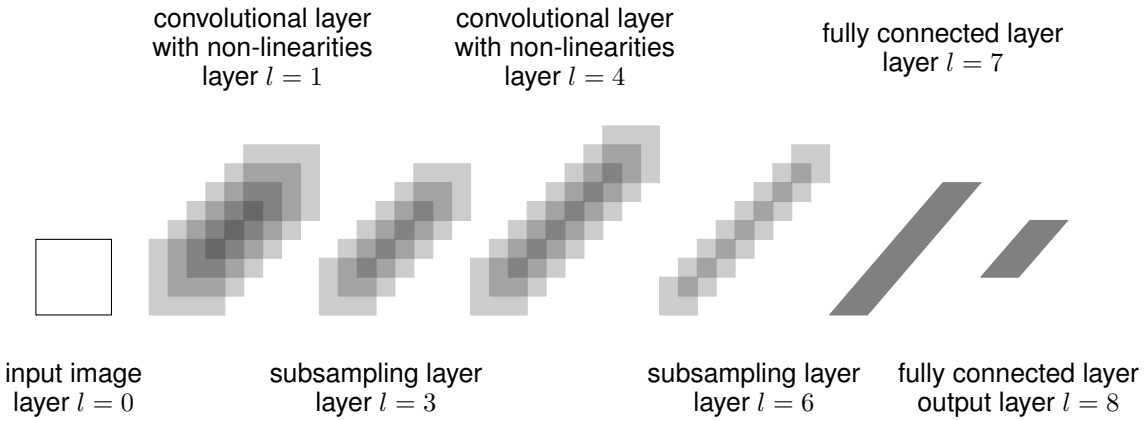

Figure 5: The architecture of the original convolutional neural network, as introduced by LeCun et al. (1989), alternates between convolutional layers including hyperbolic tangent non-linearities and subsampling layers. In this illustration, the convolutional layers already include non-linearities and, thus, a convolutional layer actually represents two layers. The feature maps of the final subsampling layer are then fed into the actual classifier consisting of an arbitrary number of fully connected layers. The output layer usually uses softmax activation functions.

What Is Fine Tuning And Why Should We Do This Finetuning means taking weights of a trained neural network and use it as initialization for a new model being trained on data from the same domain (often e.g. images). It is used to:

- Speed up the training
- Overcome small dataset size

There are various strategies, such as training the whole initialized network or "freezing" some of the pre-trained weights (usually whole layers).We do that except last layer.

Why do we freeze the rest and train only fc layers

- Computation time: If we are not freeze all the layers there will be some consequences. Such as backpropagating and updating the weights all the layers of the network, this means a huge decrease in computation time. For this reason, if you unfreeze all the network, this will allow you to see the data fewer epochs than if you were to update only the last layers weights'. In this situation we just unfreeze last layer and backpropagation process just apply on last layer. So that computation time is not grow too much and accuracy balance not decrease high amount.
- Accuracy: Of course, by not updating the weights of most of the network your are only optimizing in a subset of the feature space. If your dataset is similar to any subset of the imagenet dataset, this should not matter a lot, but, if it is very different from imagenet, then freezing will mean a decrease in accuracy. If you have enough computation time, unfreezing everything will allow you to optimize in the whole feature space, allowing you to find better optima.

SOFTWARE USAGE:

In this part i will use pre trained model and freeze all layers then start train network. Before starting train process we prepare network for training process. That means we have 10 classes and pre trained network has 1000 classes for last layer. We must change this fc8 layer like 4096x10 dimension. After do that initilaze optimizer. We must use ADAM optimizer in this example. For this optimizer i decide best learning rate value is 0.0001. This gives me best result.Assign criterion as Cross Entropy Loss and assign a scheduler. Then start training. After the training we change model to evaluation mode and evaluate test images response then calculate accuracy for best.

EXPERIMENT RESULTS:

In this part i will describe training process and test process seperately. In training process train/valid loss graph will be explained. Then comment about overfitting network.

Train Mode

Batch Size:16 Learning Rate:0.0001->

```
Training completed in 25m 30s
Best acc: 0.8520
```

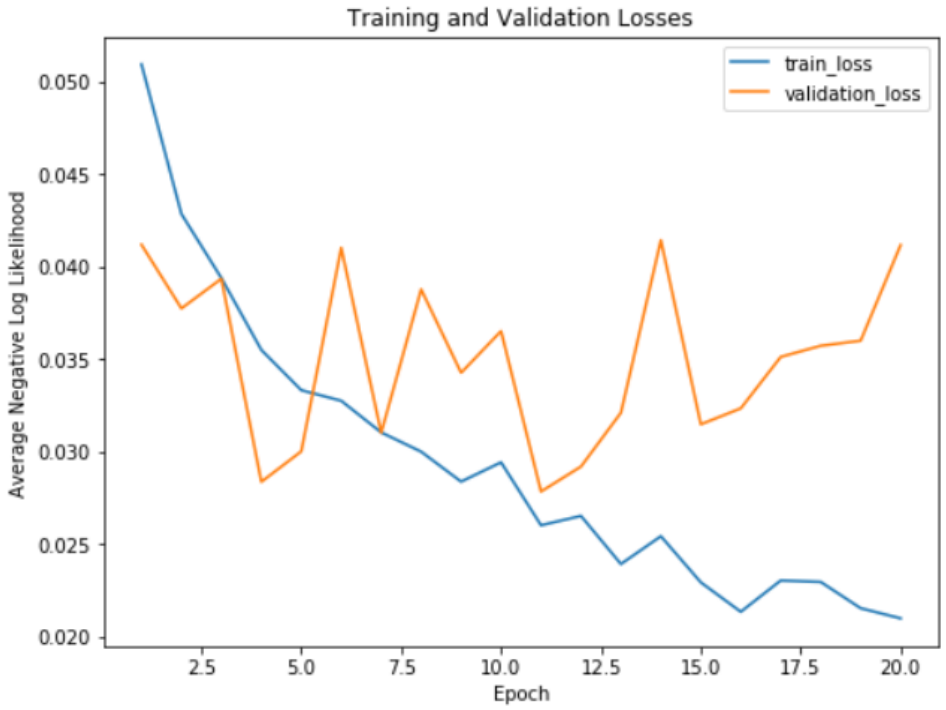

Figure 6: Loss Graph

Epoch 10/20

Training batch 200/250
Validation batch 0/16
Epoch 10 result:
Avg loss (train): 0.0260
Avg acc (train): 0.8560
Avg loss (val): 0.0279

Avg acc (val): 0.8520
———-

This graph shows while network training validation loss in decreasing in some part but not informaly. After 10-12.5 epoch network starts overfitting that means train mode must stop now but in this experiment i don't implement early stopping because of this graph. Some part valid loss decreasing and increasing so we can't stop immediately at any point. After the 20 epoch i choose best network checkpoint and save them if network getting overfitted it doesn't matter.

Batch Size:32 Learning Rate:0.0001

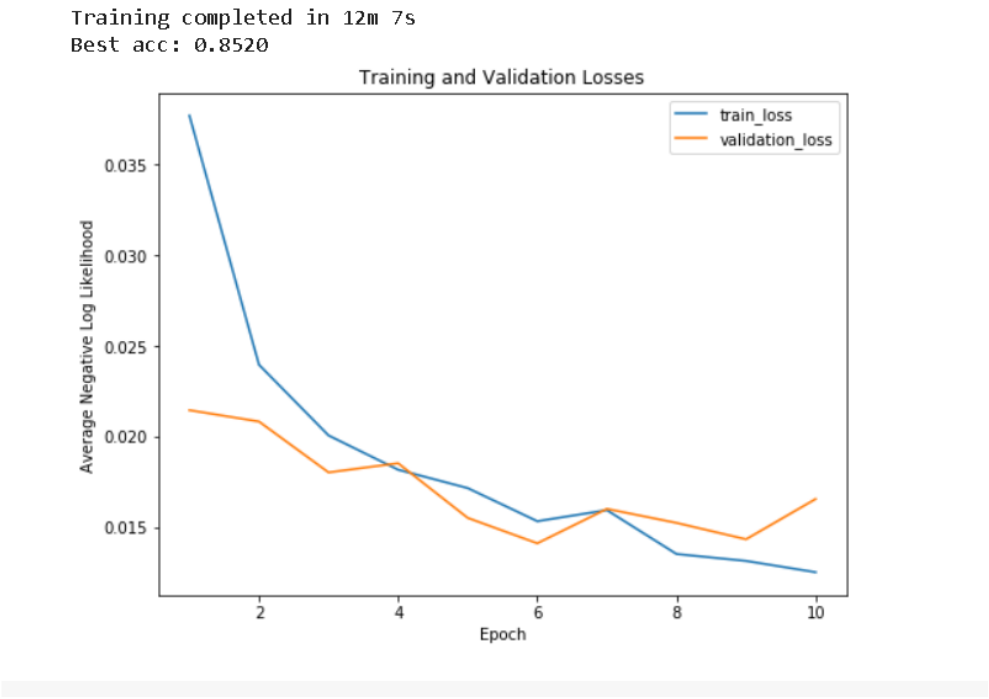

Figure 7: Loss Graph

Epoch 9/10
———-
Training batch 125/125
Validation batch 8/8
Epoch 9 result:
Avg loss (train): 0.0145
Avg acc (train): 0.8720
Avg loss (val): 0.0153
Avg acc (val): 0.8520
———-

This graph shows more clear about process of training. In starting position we see that program underfitting. While train continue until 8-10 epoch training process going succesfully. After that valid loss start increasing and that means network getting overfitted. Then process stops.

Test Mode

After a couple of test i decide best learning rate as 0.0001. We see that in my model while 16 batch size is better than 32 batch size. In contrast to this running time parameter is better than other one.

Batch Size	Learning Rate	Average Loss	Average Accuracy
16	0.0001	0.0243	%85.20
32	0.0001	0.0147	%87.60

Implementation Details:

In this part there is some explanation about implementation steps:

- Collect all images in dataloader.
- Create vgg16 pretrained model and download it.
- Freeze all layers in model except last one.
- Initiliaze training parameters optimizer(ADAM),scheduler,criterion(CrossEntropyLoss)...
- Call train model functions.
- Train network with using training and validation images.
- After training is done evaluate model on test images and calculate accuracy.

4 Part 3

In the third part we do the same job in part 1. Because of that there is no explanation again about implementation details or software usage. Different from part 1, in this part we use fine tuned model which trained in part 2 for extract features.

EXPERIMENT RESULTS:

Table 2: Part 3 results

Part		
Batch Size	Correct	Accuracy
16	211	%84.4
32	221	%88.4

When compare this results to part 1 results. We see that results changed. The reason of this we trained again this pre trained network with our train and validation images. And this images close to our test images. So this model become more suitable for our prediction process.

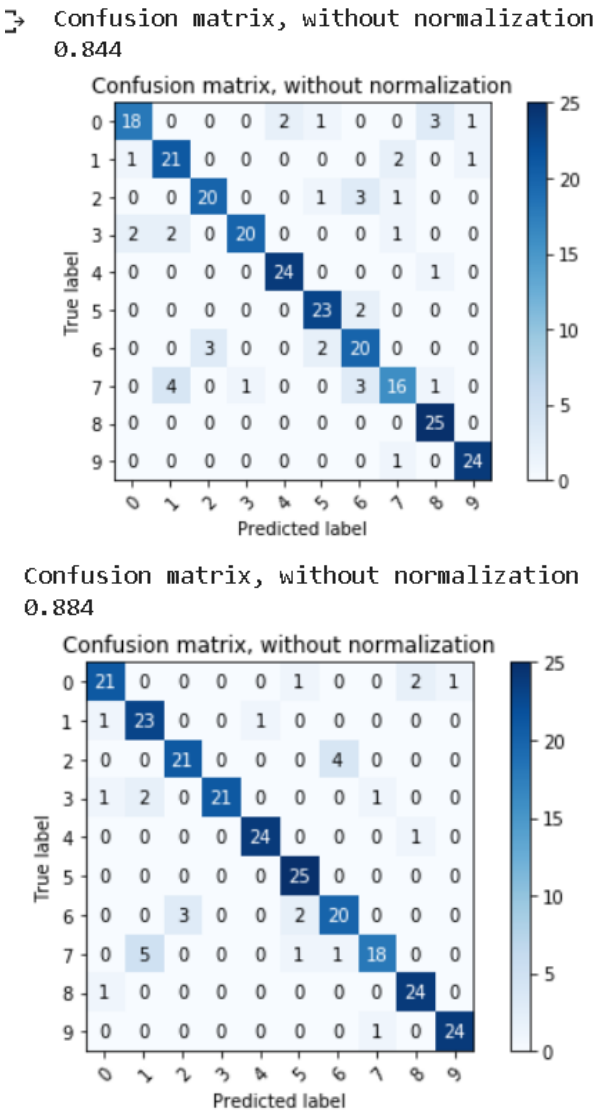

Figure 8: Confusion Matrices

5 Functions:

plotConfusionMatrix(): This function draw confusion matrix with svm results.

extractFeature(): Extract fc7 features from given pre-trained model and phase.

getAccuracy(): Return svm classification accuracy.

visualizeModel(): Generic function to display predictions for a few images

evalModel() Evaluate accuracy about test images on training network.

trainModel() Train on pre trained network.

For other process built-in functions used.

6 Conclusion:

In conclusion we understand that learning rate and batch size values are important in transfer learning. We test two batch size and both of them give different result. In situation depends one which one of must be used. When i create firstly this experiment i used Stochastic Gradient Descent as optimizer.Then i used learning rate 0.001 value. But when convert them to the ADAM optimizer this value is too big for it.Then i change learning rate 0.0001 and this is the best option for this optimizer in my experiment.